Position Paper for the Exascale Operating and Runtime Systems Workshop

Jack B. Dennis
MIT Computer Science and Artificial Intelligence Laboratory
dennis@csail.mit.edu

July 10, 2012

Preamble

- There is much current interest in developing runtime software for massively parallel computer systems.
- Much of this work exposes areas of inefficiency that could be significantly reduced by changes in system, memory and processor architecture.
- There is increasing interest in systems that relieve the user of responsibility for managing processor deployment and movement of data between levels of memory. This interest arises from the new crop of applications that have irregular structure and are ill-sited to preplanned resource mapping.
- Such applications need the benefits of a computer system that manages memory and processing resources dynamically in response to needs of the application.

Program Execution Model A Program Execution Model (PXM) is a precise specification of the interface between application code and all aspects of the computer system (hardware, runtime, operating system) on which the application depends for its correct operation.

Position Statement Research on runtime systems should strive to develop and demonstrate a PXM to specify the processor/memory architecture of a new generation of massively parallel computer systems.

CAUTION It is easy to fall into the trap of sacrificing the goal of a PXM for the future to achieve performance on today's inadequate hardware, thereby failing to create value for future system architects.

Approach The reasoning sketched below has motivated our work toward architecture of systems employing hardware implementation of dynamic resource management. This can serve as a guide for the design of runtime systems that cold make a genuine contribution to architectural choices for exascale computing.

- The units of both memory and processing resource allocation must be small to permit spreading computations over the resources of a massively parallel computer.
- Operating system and runtime software executes processor instructions thereby consuming energy and sapping performance.
- Therefore, applying hardware technology to operations that require large numbers of non productive instruction executions should reap significant benefits in energy efficiency and performance. One obvious candidate for hardware implementation is task scheduling.
- In today's computer systems task switching is a needlessly costly operation. This is due mainly to the change of addressing environment required when shifting a processor between unrelated tasks.
- This cost can be greatly reduced by arranging that all tasks running on the system operate in the same addressing environment. This implies use of global address space and virtual memory.

- Virtual memory has been the most successful advance in computer architecture in support of programmability. Most of us are unaware of the programming challenge posed by the early machines with limited main memory and no paging.
- What is needed now is an integrated virtualiztion of both processing and memory resources for massively parallel systems.

Architectural Innovation for Managed Resources The Fresh Breeze project [1] has grown from a conviction that the qualities desired for a massively parallel computer system can only be achieved through fundamental changes to conventional system architecture, especially in the area of memory system structure and access, as well as providing for efficient fine-grain scheduling of tasks. These considerations have led to the definition of a new program execution model (PXM) for massively parallel computation – one that satisfies all six principles for supporting modular software construction [2]. The PXM uses tiny tasks as the scheduled units of computation and fixed-sized “chunks” of memory as the units of memory allocation and management. Recent simulation experiments conducted at University of Delaware have demonstrated promising performance for linear algebra computations[3, 4] and the breadth first search problem[5]. The capability for performing more accurate simulations for larger system models is currently under development.

The Burroughs B5000 and Successors Around 1961, the Burroughs Corporation began producing computer systems in the series that became known as the Burroughs B5000 series. These were the first computer systems designed for operation in multiprocessor configurations. Four processor cabinets could be interconnected with as many memory cabinets in what we now call a symmetric shared memory architecture. The instruction set was designed so programs were stored as read-only code, making it possible for program parts running on different processors to execute the same code simultaneously. Burroughs provided the Automatic Scheduling and Operating System (ASOP) which was loaded into shared address space and could be executed concurrently by all processors. The ASOP provided resource management services to concurrent users of the computer system.

The *descriptors* of the Burroughs systems provided a form of segmented virtual memory. A descriptor located a *segment* of memory, giving its base memory address and its length. All user, and much system, code and data were stored in segments of memory and accessed only through descriptors. Being implemented in hardware, performance was at memory speed and it was impossible for a user to circumvent its protection features. Use of descriptors provided a degree of safety and protection absent from all of today’s popular computing systems.

For multiprocessing, the ASOP supported the spawning of a hierarchy of tasks by a user program. Because identifiers of descriptors are locations in the shared address space, they could be passed to tasks running on other processors and still be effective for accessing data. This way of structuring parallel programs supports a general and efficient pattern for writing independent software components.

The B5000 design implements a PXM that deserves study, extension and application to architecture for future massively parallel computation.

Qualities of the Approach An assessment of the approach along the suggested dimensions:

- **Challenge Addressed** Programmability through a programming model that supports modular program construction. Energy efficiency by replacing software with hardware implementation of fundamental resource management mechanisms.
- **Maturity** The Fresh Breeze memory model has been simulated for up to 40 processors for algorithms as varied as matrix multiplication and breadth first search.
- **Uniqueness** Use of hardware implemented virtualization of processing and memory is unique.
- **Novelty** Write-once memory model using trees of chunks; integrated hardware implementations of memory management and task scheduling.
- **Applicability** Applicable over the range of general purpose parallel computer systems.
- **Effort** An accurate simulation for systems with hundreds or thousands of cores is under development. The processor core architecture is an augmentation of simple RISC architecture.

References

[1] J. B. Dennis, "Fresh Breeze: a multiprocessor chip architecture guided by modular programming principles," *ACM SIGARCH Computer Architecture News*, vol. 31, no. 1, pp. 7–15, 2003.

[2] J. B. Dennis, "A parallel program execution model supporting modular software construction," in *Massively Parallel Programming Models*, pp. 50–60, IEEE, 1997.

[3] J. B. Dennis, G. R. Gao, and X. X. Meng, "Experiments with the Fresh Breeze tree-based memory model," in *International Symposium on Supercomputing, Hamburg*, June 2011.

[4] J. B. Dennis, G. R. Gao, X. X. Meng, B. Lucas, and J. Slucom, "The Fresh Breeze program execution model," in *Parallel Computing, Ghent, Belgium*, August 2011.

[5] T. S. John, J. B. Dennis, and G. R. Gao, "Massively parallel breadth first search using a tree-structured memory model," tech. rep., University of Delaware, 2012. Paper presented at PMAM 2012.

[6] E. I. Organick, *Computer System Organization: The B5700/B6700 Series.* Academic Press, 1973.